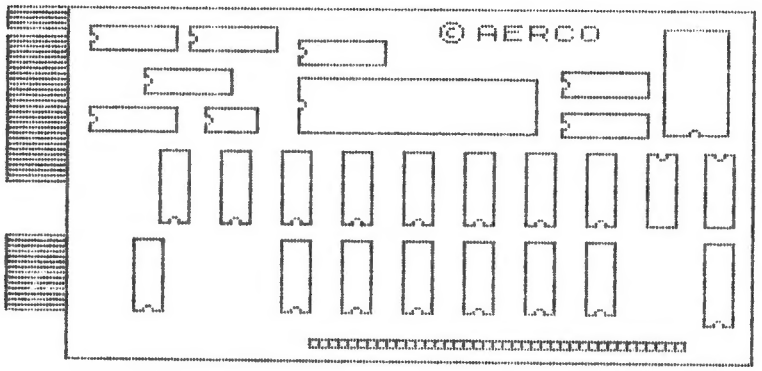
JULY 1986 VOL. 1 ISSUE 2

THE FD-68 USER



>>>>>>>>> From the Editor's Desk <<<<<<<<<

Well, here it is. The second issue of The FD-68 User. I'll bet you thought I'd been kidnapped by Gypsies or something! I'd really like to thank all of you for your patience. I had to have written the final copy at least a dozen times. I kept getting more ideas and finally had to draw the line somewhere.

The response to the newsletter has been great. I've enjoyed talking with many of you on the phone in the past few months. I must say that there is no lack of interest on the part of readers. I started out by mailing 12 copies. This second issue is going out to roughly 8 times that number. I'd like to give a special thanks to Aerco for the copies they have mailed out for me. A thanks also to those publications which mentioned this newsletter. Every little bit has helped.

The buzzword these days is CP/M. This recent addition from Aerco opens up a whole new world to FD-68 users. The more I learn about CP/M, the more excited I get! There seems to be no end to what you can do with it. There are literally thousands of public domain programs out there. I've included some general information about the new operating system.

As before, any articles, programs, or comments (good or bad) are welcome. If you send an article, Tasword II or Mscript files are fine. If you don't have a word processor and still have something to contribute, write it down and send it in. If you send a program, it would be a great help if you could include your phone number if possible. Sometimes I can't get things to work properly and a short conversation can often clear the matter up.

Some things in the works for the next newsletter. Pro-File, up to 60K file length. It only works on paper so far. Stay tuned. A reader here in Michigan has not only put OS-64 in the Dock bank, but has it working with the disc as well. One catch though, you will need the 256K upgrade configured for the 4 drive set up (you don't necessarily have to have 4 drives). This same reader has sucessfully accessed all of the 256K and is willing to share this information with the rest of us. I don't have the information yet myself or it would be in this newsletter. To insure that you receive future issues, see the last page for subscriber information.

I've received a number of comments from readers about the time it takes for new developments to materialize from their favorite companies. We need to keep in mind that we aren't dealing with companies who have teams of engineers busily working toward the ultimate in software and hardware. Most are smaller operations which make the best use of available time and resources to develop new products.

When Timex pulled out, not only did they leave all of us users high and dry, they also left all of our suppliers holding the bag. Most of them have folded. Myself, I am very grateful for the existence of those companies that have remained. I would not be writing this newsletter if they had all given up. The number of new products which are made available for a computer that was abandoned 2-1/2 years ago by its manufacturer never ceases to amaze me. As users, we should be supporting, not criticizing them. No sense in bitting the hand that feeds us. This support goes both ways. We need them, and they need us. Niether will survive long without the other.

Well, instead of listening to me, I'll let you get to the material inside. I hope you enjoy it.

The hardware modification for copying cartridges to disc is now available from AERCO. You can make the mods yourself as shown in the illustration, or for $15 AERCO will send you the assembled chip and switch and all you have to do is install it. Once the hardware change has been made, proceed as follows:

1) Turn off the computer
2) Slide the switch to the UP position
3) Insert the cartridge you wish to copy
4) Turn the system back on and MOVE "name.aro",
5) Turn off the system
6) Remove the cartridge
7) Slide the switch to the DOWN position
8) Turn on the computer and CAT "name.aro",

The program will load and run. I have the hardware change on my interface and it works well, to a point. After some trial and error, I found that I couldn't copy all of the cartridges I had. The only ones I could copy successfully were Androids and Crazybugs. I couldn't get Flight Simulator, States and Capitols, Pinball, or Home Budgetter to copy. A talk with Jerry at AERCO cleared up the matter. It seems that when the .ARO extension is called in its' present form, it only copies 16384 bytes, 32768 to 49152, chunks 4 & 5 of the dock bank. Evidently the other programs are longer than that. Jerry said that a change in the EPROM code, would read the header from the cartridge, and save the number of bytes accordingly.

This hardware modification will not copy an LROS cartridge such as OS-64. The .ARO extension only copies from 32768 to 49152. An LROS (language oriented software) program resides at memory addresses 0-32768.

I would be interested in hearing from those of you who may have had similar experiences.

HELP!

This regular column will be provided for those of you with questions or problems with software or hardware. If you can help, drop them a line.

1) Anyone knowing how to use an Olivetti PR2300 printer with Tasword II please contact:

   Arthur Adams
   18201 20 Mile Rd.
   Marshal, MI 49068

2) Dale has a Tasman C printer interface and says it works fine with all programs except Tasword II. If you can shed some light on the problem write to:

   Dale Fitchie
   3444 East 40th St.
   White Cloud, MI 49349

3) Dan has been having difficulties using Fig FORTH with the disc. He has successfully transferred it to disc but is experiencing operating problems once it has loaded.

   Daniel Callison
   1716 Wishing Well Way
   Tampa, FL 33619

NYBBLES AND BITS

In the first issue I asked for information on the possibility of more than one ROM being issued for the 2068. From the information I've received from readers, there appears to have been only one ROM ever issued. If you PRINT PEEK 19 you should get 255 regardless of when your machine was made.

****

Other sources of information of interest to Aerco FD-68 users:

SyncWare News P.O. Box 64 Jefferson, NH 03583	The Sinc Times 1707 King St. Jacksonville, FL 32204
Time Designs Magazine Co. 29722 Hult Road Colton, OR 97017	Jack Dohany 325 O'Connor St. Menlo Park, CA 94025

****

There is a new bulletin board in Atlanta GA. that has an Aerco FD-68 sub-board. Its' name is Bill's Obsession, run by William Ericson. The hours are 1 A.M. to 8 A.M. guaranteed. The system is up roughly 90% of the other hours. It has a good message base and also offers uploading and downloading capabilities for both the 2068 and RP/M formats. You can leave me a message on the board for me if you have questions regarding the newsletter.

Phone (404) 377-2550

****

One of the many things that Aerco is working on is the .LRO extension. This requires moving the DOS to chunk 6 (which is more involved than you may think). Jerry has a working version of it but says it needs to be cleaned up some before it is released. This would enable us to use the Spectrum in the Dock bank with disc capabilities. Maybe even with OS-64? Time will tell.

****

The most recent version of the FD-68 EPROM is 8.9. If you don't have this newest version, send $5.00 to cover the cost of the EPROM (includes S&H) to Aerco and they gladly send you the update.

****

DUAL-SCREEN MODE

This article is reprinted from the Capitol Area Timex User's Group. It is written by H.E. Weppler.

I have been intrigued by the idea of the Dual-Screen Mode as mentioned in the 2068 User Manual (p.248). That uses a second display file which may be shown on the screen instantaneously on command. The simplified M/C program in Figure 1 will set that up. It may be used for menus, instructions or graphics which are to be called up repeatedly by a program. But it also gives us an opportunity to explore more of the functions of the 2068. The SET-UP portion of the program first switches to the Extension Rom (EXROM). There it calls up the CHN6_V routine, which rearranges the memory area and establishes the second display file (see p.254). After that it switches back to the Home ROM. DF1-TO-DF2 copies the contents of Display File 1 into Display File 2.

To load the M/C, run the program in Figure 2, manually entering the decimal inputs shown and SAVE the CODE for future use. Before using, CLEAR 63199. Enter RAND USR 63200 to switch from the normal case (Mode 0) to Mode 128, in which the two display files are available, with Display File 1 (DF1) on the screen. The screen may then be filled from the keyboard or memory, as usual. When it is desired to store the DF1 contents in DF2, enter RANDOMIZE USR 63224. DF1 is still showing on the screen, which may be cleared and loaded with new material. To display DF2 (Mode 1) just enter OUT 255,1. To return to DF1 to the screen (Mode 128) enter OUT 255,128. But one note of caution -- when in Mode 1 (DF2) any keyboard inputs or error messages will go to DF1 and will not be seen on the DF2 screen. A command to return to DF1, such as OUT 255,128 will still be effective but will not be seen and so must be entered very carefully to avoid error. It is preferable for a Mode 1 command to include a return instruction. For example, with OUT 255,1: PAUSE 0: OUT 255,128 the return to DF1 will be completed accurately when any key is pressed.

Some alternatives:

1) To return to the one-display mode (Mode 0) and restore the memory areas to normal, enter RAND USR 63204. However, this is not normally needed. Once Mode 128 is in use, all the functions may be used with DF1, Whether or not DF2 is used.
2) If there is occasion to transfer a display from DF2 to DF1, possibly to revise a part of DF2 or to copy DF2, enter RAND USR 63225. But note that this is not needed to just put DF2 on the screen. That is done with OUT 255,1.
3) As a stunt, with 2 messages loaded into DF1 and DF2, you might enter OUT 255,6 or OUT 255,62 to demonstrate the "64-column mode" which puts both displays on the screen at once, with the characters interweaved.
4) To COPY DF2 without disturbing DF1 use the following short M/C program.

```
COPY DF2    DI
            LD B,176
            LD HL,24576
            CALL 2568
            RET
```

This redirects the printer to the start of DF2 and then calls in the regular printer routine. POKE the 10 bytes starting at 63246 with: 243,6,176,33,0,96,205,8,10,201. For COPY DF2 enter RAND USR 63246. This may be done with DF1 on the screen.

H.E. Weppler

FIGURE 1. M/C program

```
SET-UP      LD A,128
            JR,1
            XOR A
            PUSH AF
            LD A,128
            OUT 255,A
            LD A,1
            OUT 244,A
            POP AF
            CALL CHN6_V
            LD (VIDMOD),A
            XOR A
            OUT 244,A
DF1-TO-DF2  AND A
            LD HL,16384
            LD DE,24576
            KD BC,6912
            JR NC,1
            EX DE,HL
            LD A,(VIDMOD)
            CP 0
            RET Z
            LDIR
            RET
```

```
CLEAR           63199
SET-UP          RAND USR 63200
DF1-TO-DF2      RAND USR 63224
MODE 128 (DF1)  OUT 255,128
MODE  1  (DF2)  OUT 255,1
```

FIGURE 2. Decimal Loader

```
10 FOR N=63200 TO 63245
20 INPUT (N);" ";B
30 POKE N,B
40 PRINT N;" ";PEEK N
50 NEXT N
```

INPUT the following: 62,128,24,1,175,245,62,128,211,255,62,1,
211,244,241,205,142,14,50,194,92,175,211,244,167,33,0,64,17,
0,96,1,0,27,48,1,235,58,194,92,254,0,200,237,176,201

BASIC IN THE DOCK BANK

Until recently I'd geared my thinking toward the use of only machine code in the Dock bank. The ability to use Basic in the Dock bank offers some interesting possibilities. Using two banks of Basic would enable each to have its own set of variables, line numbers, defined functions, etc. The same variable names can be used in each bank and not cause a conflict, because each Basic program operates independant of the other.

Using Basic in the Dock bank requires some general knowledge of how the Timex system works. It may be helpful to refer to page 255 of the 2068 owners manual. This shows how the Timex System is structured. On fire up, the computer divides the memory as shown in Figure C-1. Memory addresses 23552-23755 are known as the System Variables. Refer to pages 262-265 of the owners manual and you will find a list of the System Variables and the function each one performs. Properly altering the System Variables is crucial to successfully writing Basic in the Dock bank. The System Variables we need to concern ourselves with are:

PROG - Starting address of the Basic program. (23635-23636)
VARS - Starting address of the Basic variables. (23627-23628)
ELINE- Address of current command being typed in. (23641-23642)

To find the start of the Basic program area, PEEK the System Variable PROG. With no program in memory, PRINT PEEK 23635+256* PEEK 23636 in the immediate mode. The result should be 26710 (with a 64 column operating system this result would change to another fixed address). This is the starting address of the Basic program area in the Home bank. To find the start of the variables area (VARS), PRINT PEEK 23627+256*PEEK 23628. The result will also be 26710. This is because there is no program in memory. As soon as a Basic line is entered, VARS will move up an amount equal to the length of the Basic line.

Again refer to page 255, Figure C-1. Take notice of the pointer with the 80h (128 dec) above it just ahead of PROG, and then again just before E_line. These mark the end of their respective memory areas. In a similar manner, the end of each Basic line is marked with a code 13 (ENTER). When the computer searches for a variable, it starts at the address pointed to by VARS and continues until it finds a match. If no match is found by the time it reaches the 80h, it will return with a Variable not found error. Therefore these markers are very important.

In order to write Basic in the Dock bank, you must alter the System Variables PROG, VARS, and ELINE. You need to POKE them with the lo and hi bytes of the new location. Simply altering these addresses is not enough. Once altered, the computer expects to find those end of area markers. Therefore, you need to POKE in these markers as part of your switch to Dock Basic.

You might think of the computer's memory as a long file drawer, and all the files are stored alphabetically (a,b,c,d, etc.). Imagine trying to find a file without knowing where the A's end and the B's begin. This is what the computer is faced with if you don't POKE in the end of area markers. First thing to do is to OUT 244,241. This enables chunks 0,4,5,6,7 of the Dock bank. Now to alter the System Variables to point to 32768 of the Dock bank, in the immediate mode, enter this as one line:

```
POKE 32768,128: POKE 32769,13: P
OKE 3277Ø,128: POKE 23641,1: POK
E 23642,128: POKE 23627,Ø: POKE
23628,128: POKE 23635,Ø: POKE 23
636,128
```

You are now in the Dock bank with your Basic starting at address 32768. Verify this by doing a PRINT IN 244, this should be 241 (signifies Dock bank), then PRINT PEEK 23635+256*PEEK 23636. If you typed the line in correctly, you should get 32768. To return to Home bank, enter as one line, the following:

```
OUT 244,1: POKE 2671Ø,128: POKE
26711,13: POKE 26712,128: POKE 2
3641,87: POKE 23642,1Ø4: POKE 23
627,86: POKE 23628,1Ø4: POKE 236
35,86: POKE 23636,1Ø4
```

Keeping Track of Your Banks

When you write Basic in either bank, the exact location of all the pointers and markers change as the program grows. Just as the starting addresses of the Basic areas are different (26710 & 32768), so are all the pointers and area markers. Once you get this working OK, you will need a way to switch between the two banks of Basic under program control and start running in the other bank at any line. In order to successfully switch from one bank to the other and return again, you need to keep track of, and store the current pointers and markers, so you can restore them to these same locations when you return from the other bank.

The System Variables keep track of many many things. There may be things in there that will effect the proper switching between banks of Basic, that we have not even considered. With that in mind, let's store the contents of the entire System Variables rather than take a chance on missing something important that would result in a crash. The System Variables are 203 bytes in length. You will need two, 203 byte areas in which to store the pointers and such, as they pertain to the Home bank, and one for the Dock bank. A good place to store the Home bank variables is in the printer buffer (as opposed to lowering RAMTOP or putting them in a REM statement). This will allow you 99.9% compatibility with any software you now have when used in the Home bank (there are always exceptions). For the Dock bank, we will move the start of the Basic area up from 32768, to 32972, which will allow for the 203 byte storage area needed for the variables. This is analogous to raising the floor instead of lowering the ceiling.

What you now have is a way in which to use two banks of Basic or Basic and machine code, switch between them under program control and have complete access to Home bank RAM. All of this comes with only a 203 byte sacrifice in the Dock bank. Granted, there are different places to store the variables and the machine code, but I feel this method allows you the greatest flexibility while using a bare minimum of available RAM.

Now enter the machine code that makes all of this possible. Type in the following Basic program and RUN it.

```
1Ø FOR a=24320 TO 24478
2Ø READ b: POKE a,b: NEXT a
```

```
  30 DATA 243,219,244,8,58,158,9
5,6,0,184,40,62,8,6,1,184,40,28,
33,0,92,17,0,128,1,203,0,237,176
,62,1,211,244,33,0,91,17,0,92,1
  40 DATA 203,0,237,176,251,201,
33,0,92,17,0,91,1,203,0,237,176,
62,241,211,244,33,0,128,17,0,92,
1,203,0,237,176,251,201,62,1,211
  50 DATA 244,50,158,95,33,0,92,
17,0,91,1,203,0,237,176,62,241,2
11,244,33,204,128,54,128,35,54,1
3,35,54,128,33,89,92,17,75,92,1
  60 DATA 83,92,62,128,119,18,2,
35,19,3,62,204,18,2,54,205,24,14
2,33,0,91,17,0,128,1,203,0,237,1
76,205,2,10,33,0,128,17,0,91,1
  70 DATA 203,0,237,176,201,0
```

Once this is RUN, make a copy to disc with MOVE "DOCKBAS.bin", 24320,160. After you have done this, delete the Basic with DELETE , . Then type in the following demo program.

```
   5 LET B= IN 244
  10 IF B=1 THEN PRINT AT 1,0;"H
OME CHUNKS ENABLED = ";B
  15 IF B=241 THEN PRINT AT 1,0;
"DOCK CHUNKS ENABLED = ";B
  20 PRINT '''"START OF BASIC ARE
A = ";PEEK 23635+256*PEEK 23636
  30 PRINT '"START OF VARIABLES
= ";PEEK 23627+256*PEEK 23628
  40 PRINT '"START OF E_LINE = "
;PEEK 23641+256*PEEK 23642
  45 PRINT AT 21,0;" PRESS ANY K
EY TO SWITCH BANKS": PAUSE 0: RA
NDOMIZE USR 24320: RUN
```

Save a copy of this to disc with MOVE "demo.bas",1. Now to initialize the routine, RANDOMIZE USR 24320.

Upon initialization, the machine code peforms the following:

1. Block moves the current contents of the Home System Variables to it's storage area in the print buffer (23296-23551).
2. Switches to the Dock bank, POKEs in the 80h's that mark the end of memory areas.
3. POKEs the System Variables to point to the new Dock bank addresses.
4. Block moves the System Variables, which now contain the Dock bank info, to the 203 byte storage area reserved below the Dock Basic (32768-32971).
5. Switches banks again to Home bank and block moves the 203 bytes previously stored in the print buffer (23296-23551, Home bank info) to the System Variables area.
6. Returns to Basic in the Home bank.

After the system is set up, you will still be in Home bank. List the program. Everything looks fine. Now, RANDOMIZE USR 24320 and LIST the program again. You will see nothing there because you are now in the Dock bank and you haven't loaded any program there. Next CAT "demo.bas",. The program will load and auto run. Take notice of the chunks enabled and the starting address of the various System Variables. Press any key. This will switch you back to Home bank and stop. RUN the basic in the Home bank. Take notice of the chunks enabled etc. Now, pressing any key will switch you back and forth between banks and start running the program.

A Little Hindsight

The method by which you exit a running program in one bank, and start running a program in the other bank at any given line, came about by accident. I thought I would try RANDOMIZE USR 24320: RUN, to see if I could switch banks and start running. Well, it did indeed switch banks OK, but it just sat there with a 0 OK,0:1. I figured that, at least I had tried. Then I did a RANDOMIZE USR 24320 to return and the program in the bank I just left automatically started running. It occurred to me that whenever you call the code to switch banks, a copy of the System Variables is made, and then reinstated upon return. These are, of course the same System Variables that keep track of the next command to be executed. When I made the first RANDOMIZE USR call, the computer acted on the code I specified, but appeared to ignore the RUN command. But when it returned there, it picked up right where it left off, which was the command RUN. The command that follows the RANDOMIZE USR 24320, should be the point at which you would like to continue running upon return from the opposite bank. Due to the fact that you are moving all the System Variables, you will notice that it will return the screen colors as they were before you left.

Printing From the Dock Bank

Using the 2040 printer from the Home bank presents no problem. While it does indeed write over the code we stored in the print buffer, (which contains the Home bank info) the code makes a new copy of the System Variables there again before exiting to the Dock bank.

In the Dock bank, using the printer is a little different. If the printer is called into service, it will overwrite the code we have stored in print buffer, which contains the info we need to successfully return to the Home Basic. To get around this, we have to write a routine that block moves the contents of the print buffer into the storage area below the Basic in the Dock bank, calls the screen copy command from the Timex ROM, and when finished, block moves the storage area below Dock Basic back into the print buffer for retrieval when we exit Dock bank and enter the Home bank. While we did write over what was in storage area below Basic, which contained the Dock bank return info, it will be restored as a part of exiting the Dock bank. To do a screen copy from the Dock bank use RANDOMIZE USR 24452 instead of COPY.

Saving and loading programs in the Dock bank.

Saving and loading Basic programs while in the Dock bank is no different than Home bank. Machine code requires special attention. When a program (or file) is saved with the .bin extension, regardless of which bank you are in, along with the program, the current value of port 244 is also saved. If a binary file is to

be loaded into the Dock bank, it must have been saved from there. This is important because the file will load back in to where it came from regardless of the current chunks that are enabled. In order to save programs from the Dock bank that presently reside in Home bank, enter the XFER bytes machine code listed below. Use the BANKMON program on the AERCO boot disc to find the starting address and the length of the code you wish to transfer. (Bankmon is available from Jack Dohany 325 O'Connor St. Menlo Park, CA 94025)

```
  10 FOR a=24320 TO 24357
  20 READ b: POKE a,b: NEXT a
  30 DATA 1,241,17,0,130,33,0,13
0,1,0,126,58,0,95,211,244,126,8,
58,1,95,211,244,8,18,19,35,11,12
0,177,32,235,58,1,95,211,244,201
```

After you have RUN the program, save it to disc with MOVE "XFER.bin",24320,58. Then delete lines 10,20, and 30 and type in the following program.

```
   5 CAT "xfer.bin",
  10 CLS : INPUT "Enter file nam
e to be transfered"; LINE a$: LE
T b$=a$
  15 IF LEN b$>9 THEN LET b$=b$(
 TO 9)
  20 LET b$=b$+"*"
  25 LET a$=a$+".bin"","": CAT "a
$",
  30 INPUT "Starting address ";a
: LET s=a: GO SUB 100
  40 INPUT "# of bytes for XFER
(max 32768) ";b: IF b>32768 THEN
 PRINT "Code too long for Dock B
ank": STOP
  45 LET s=b: GO SUB 110
  50 RANDOMIZE USR 24322
  60 LET b$=b$+".bin"","+STR$ a+
","+STR$ b: MOVE "b$",
  99 OUT 244,1: STOP
 100 GO SUB 200: POKE 24326,lo:
POKE 24327,hi: POKE 24323,lo: PO
KE 24324,hi: RETURN
 110 GO SUB 200: POKE 24329,lo:
POKE 24330,hi: RETURN
 200 LET hi=INT (s/256): LET lo=
s-(hi*256): RETURN
```

Save this program with MOVE "transfer.bas",5.

This program will first load in the XFER machine code and ask for the name of the .bin file you wish to transfer. You will then be asked for the starting address and length of the code. Then the program will be transfered to the same address in the Dock bank and saved to disc as a Dock bank file. An asterisk will be added to all files that are saved from the Dock bank. In this way, it will be a simple matter to determine which are Home bank files and which are Dock bank files while looking at the disc directory.

This concludes the Basic in the Dock bank. While I can't guarantee that all these routines will work in every situation, I have given them a thorough thrashing with good results. It should keep you busy for a while.

I'd like to thank Tom Bent of SYNCWARE NEWS for steering me the right direction. He planted the seed that grew into this article.

Alternative Power Supplies:
by Leland Fiscus

I purchased a switching power supply from JDR Microdevices, Part# PS-11951, for $29.95. It is made by ASTEC, and has outputs of +5V @ 6A, +12V @ 2A, +12V @ 1.5A, and -12V @ .2A. The only thing wrong is that the unit comes with absolutely NO documentation. Maybe I can give some pointers in dealing with this unit.

If you look at the component side of the unit, with the transformer "HIGH VOLTAGE" label on the left, you will see 3 terminal strips. At the top, there is a strip with 8 pins labeled with the output voltages. On the right side there is a strip with 4 pins labeled "L" and "N". On the bottom there is a strip with three pins marked "115" and "230". The bottom right mounting hole has an earth-ground stenciled next to it.

First, foremost, and always---NEVER ground the bottom right mounting screw of the power supply if your drives are grounded to the disc enclosure. The ground is meant only for the input filter section, and has enough potential to draw sparks from the system ground. In fact, I recommend using a shoulder washer, similar to those used to mount power transistors, from the bottom side of the board to make sure that the ground trace remains isolated from the rest of the circuitry.

The input power for the unit is connected to the pins marked "L" and "N". The "hot" wire goes to "L" and the neutral to "N", but I have tried it, and there doesn't seem to be any difference in performance if the wires are switched. Also, for use with 115V power, a jumper is necessary between the pins on the bottom connector that are marked "115".

I have used AERCO's suggestion of using pins 2 and 34 of the disc connector to provide power to the computer, and so far it has worked very well. I use the +5V and the +12V @ 2A for the disc drives, and the +5V and the +12V @ 1.5A for the computer. I can't use the power switch on the computer to turn it on and off, but I also don't have to bother with the power transformer or finding a place to plug it in.

I also have jumpered the SOUND pin on the expansion plug to pin 2 of the RGB video connector on the FD-68 interface, and have found this to be very useful, as I can use a single cable from the interface to my monitor and have video and audio. (I wonder if Jerry ever thought of this?)

BASIC IN THE DOCK BANK

```
ADDR  HEXCODE MNEMONIC        NOTES

5F00  F3      DI              ;Disable interrupt
5F01  DBF4    IN A,(F4)       ;Fetch current chunks enabled
5F03  08      EX AF,AF'       ; and store in alternate accumulator

5F04  3A9E5F  LD A,(5F9E)     ;Check if Dock basic is initialized
5F07  0600    LD B,00         ; if address 24478=0 then initialize at
5F09  B800    CP B            ; 5F4A (24394)
5F0A  283E    JR Z,5F4A       ; If 24478=1 then continue

5F0C  08      EX AF,AF'       ;Where to now? If current bank=Home
5F0D  0601    LD B,01         ; then goto 5F2E (24366)
5F0F  B8      CP B            ;If current bank=Dock
5F10  281C    JR Z,5F2E       ; then continue
```

LEAVE DOCK BANK

```
ADDR  HEXCODE MNEMONIC        NOTES

5F12  21005C  LD HL,5C00 ;Load HL with start of system variables
5F15  110080  LD DE,8000 ;Load DE with start of sys vars storage
                         ; area for Dock bank (32768)
5F18  01CB00  LD BC,00CB ;Load BC with # of bytes to move
5F1B  EDB0    LDIR       ;Block move sys vars to 32768 Dock bank
5F1D  3E01    LD A,01    ;Enable only chunk 0 of the Dock bank
5F1F  D3F4    OUT (F4),A ; OUT 244,1
5F21  21005B  LD HL,5B00 ;Load HL with start of sys var storage
                         ; area for Home bank (print buffer)
5F24  11005C  LD DE,5C00 ;Load DE with address of sys vars
5F27  01CB00  LD BC,00CB ;Load BC with # of bytes to move
5F2A  EDB0    LDIR       ;Block move print buffer to sys vars
5F2C  FB      EI         ;Enable interrupt
5F2D  C9      RET        ;RETurn to basic
```

LEAVE HOME BANK

```
ADDR HEXCODE  MNEMONIC        NOTES

5F2E 21005C   LD HL,5C00 ;Load HL with start of sys vars
5F31 11005B   LD DE,5B00 ;Load DE with start of print buffer
5F34 01CB00   LD BC,00CB ;Load BC with # of bytes to move
5F37 EDB0     LDIR       ;Block move sys vars to print buffer
5F39 3EF1     LD A,F1    ;Enable chunks 0,4,5,6,7 Dock bank
5F3B D3F4     OUT (F4),A ; OUT 244,241
5F3D 210080   LD HL,8000 ;Load HL with start of Dock sys vars
5F40 11005C   LD DE,5C00 ;Load DE with start of sys vars area
5F43 01CB00   LD BC,00CB ;Load BC with # of byte to move
5F46 EDB0     LDIR       ;Block move stored Dock system
                         ; variables back home to 23552
5F48 FB       EI         ;Enable interrupt
5F49 C9       RET        ;Return to basic
```

INITIALIZATION

```
ADDR  HEXCODE MNEMONIC        NOTES

5F4A  3E01    LD A,01    ;Switch to Home bank and set the
5F4C  D3F4    OUT (F4),A ; initialization test byte
5F4E  329E5F  LD (5F9E),A; (24478) to 1

5F51  21005C  LD HL,5C00 ;
5F54  11005B  LD DE,5B00 ;
5F57  01CB00  LD BC,00CB ;   Before leaving the Home bank, this
5F5A  EDB0    LDIR       ;   code block moves the current system
5F5C  3EF1    LD A,F1    ;   variables to the print buffer. It
5F5E  D3F4    OUT (F4),A ;   then switches to the Dock bank and
5F60  21CC00  LD HL,00CC ;   puts in the markers used by the
5F63  3680    LD (HL),80 ;   system and alters the system
5F65  23      INC HL     ;   variables to point to these new
5F66  360D    LD (HL),0D ;   addresses. When completed, you are
5F68  23      INC HL     ;   still in the Dock bank, so it jumps
5F69  3680    LD (HL),80 ;   to the "Leave Dock" code at  5F12H
5F6B  21595C  LD HL,5C59 ;   and block moves the newly altered
5F6E  114B5C  LD DE,5C4B ;   system variables to the storage
5F71  01535C  LD BC,5C53 ;   area below the basic and reinstates
5F74  3E80    LD A,80    ;   the Home variables from the print
5F76  77      LD (HL),A  ;   buffer and RETurns to basic in the
5F77  12      LD (DE),A  ;   Home bank.
5F78  02      LD (BC),A  ;
5F79  23      INC HL     ;
5F7A  13      INC DE     ;
5F7B  03      INC BC     ;
5F7C  3ECC    LD A,CC    ;
5F7E  12      LD (DE),A  ;
5F7F  02      LD (BC),A  ;
5F80  36CD    LD (HL),CD ;
5F82  188E    JR 5F12    ;
```

SCREEN COPY FROM DOCK BANK

```
ADDR HEXCODE  MNEMONIC        NOTES

5F84 21005B   LD HL,5B00 ;Load HL with start of print buffer
5F87 110080   LD DE,8000 ;Load DE with start of Dock storage
5F8A 01CB00   LD BC,00CB ;Load BC with # of bytes to move
5F8D EDB0     LDIR       ;Block move sys vars to Dock storage
5F8F CD020A   CALL 0A02  ;Call screen copy routine from ROM
5F92 210080   LD HL,8000 ;Load HL with Dock storage address
5F95 11005B   LD DE,5B00 ;Load DE with address of print buffer
5F98 01CB00   LD BC,00CB ;Load BC with # of bytes to move
5F9B EDB0     LDIR       ;Block move stored sys vars back into
5F9D C9       RET        ; print buffer and RETurn to basic

5F9E 00       NOP        ;Storage byte for initialzation check
```

TASBANK II

Here is the next installment of Tasword II I made mention of in the first newsletter. I've received many suggestions and ideas on the program. As I stated on the last page of the first newsletter, I hoped that if nothing else, the information contained in it would spark other ideas for projects outside the uses put forth in the text. In that sense, the article achieved the desired results. It got you thinking.

Unlike the first version of Tasbank, this new version allows more flexibility in the transferring of files. As in the first version of this machine code, Dave McNeely is responsible for about 95% of XFER BYTES. There are 4 m/c routines now being used within Tasword. The routine labeled XFER BYTES, can be used as a separate routine for use with your own programs, as can the READ DRIVE and SELECT DRIVE routines. The RESET CODE is only of use within this particular program. Making use of these routines outside of Tasword is explained in the section entitled "Building Blocks".

The new menu selections are "x" for file transfer, "b" to toggle the read bank, and "d" to change the active drive. Using the "b" and "d" options are rather self explanatory. Simply pressing either key will demonstrate their use.

When you choose "x" for file transfer, you will be asked which bank is to be the source bank. This will be the bank that contains the file information you wish to transfer. Enter "d" or "h". You will now be asked for the starting line and last line of text in the source file you wish to transfer. At this point, before any transfer takes place, the length of text you are transferring is added to the current length of the destination file. If that total exceeds 19200 bytes, you will again be asked for the last line number. (There are endless additions that could be made to this such as displaying the highest line number you could enter and still be within the memory limits etc., but I'll leave those extras to you.) You will then be asked if you wish to "a" add, or "i" insert the text into the other file. If you choose "a" to add, the selected lines from the source bank will transferred and added, one line after the end of the file in the destination bank. If you choose "i" for insert, you will be asked for the line number in the destination file where you wish to insert the text. If you were to enter 53, then before the text is inserted there, everything currently in the destination bank from line 53 on, will be moved down out of the way, an amount equal to the length of the incoming text. Below are some of the variables and subroutines used in the Basic.

t = 33280 - starting address of all Tasword files
t1= 19200 - this is the maximum allowable file length per bank
t$= source bank, either "d" or "h"
s1= starting address of file to be transferred
s2= length of the file to be transferred
n = address in opposite bank where the file will be transferred
h1= length of Home bank file
d1= length of Dock bank file

8200 - activates the XFER machine code
8620 - this line figures the hi and lo bytes
8700 - POKEs hi and lo bytes of the source address
8710 - POKEs hi and lo bytes of the file length
8720 - POKEs hi and lo bytes of the destination address

The first thing you will need to do is to create the machine code. Type in and RUN the following Basic program.

```
 10 FOR a=24320 TO 24462
 15 READ b: POKE a,b: NEXT a
100 REM >>> xfer bytes <<<
105 DATA 1,241,17,0,130,33,0,13
0,1,0,126,58,0,95,211,244,126,8,
58,1,95,211,244,8,18,19,35,11,12
0,177,32,235,58,1,95,211,244,201
110 REM >>> read drive <<<
115 DATA 219,244,8,62,3,211,244
,58,237,63,230,15,95,62,8,1,68,0
,187,40,5,203,47,11,24,248,8,211
,244,201
120 REM >>> select drive <<<
125 DATA 219,244,8,62,3,211,244
,58,237,63,30,184,167,48,20,230,
15,203,39,79,58,237,63,230,240,7
1,121,176,50,237,63,8,211,244,20
1,62,1,24,236
130 REM >>> reset code <<<
135 DATA 62,1,211,244,50,0,95,6
2,241,50,1,95,175,50,3,95,50,6,9
5,50,9,95,62,130,50,4,95,50,7,95
,62,126,50,10,95,201
```

After it has run, you may want to save a copy of it to disc. Next, delete the Basic with DELETE ,. Without turning off the computer, load in Tasword II and exit to Basic.

In order to make use of the added features now included in Tasword II, extensive changes have been made to the Basic. Due to the limited memory available for Basic, some sacrifices had to be made. These changes have been made over a period of 6 months. I have included a listing of the areas of the Basic th have been changed. Line numbers 0-200 have been changed. You should alter your Basic to read just as listed below. No changes were made that I can recall between line 210 and 399. All line numbers from 500 to 2999 have undergone numerous changes as well. Make sure your Basic reads exactly as listed below. No exceptions. Lines 3000 to 6999 have been unaltered. Everything beyond line 7000 is new Basic. These lines are what manipulate the XFER BYTES machine code. Type them in very carefully to avoid errors. If you have any problems, just get ahold of me (not by the neck, please) and I'll fix you right up.

```
 10 CLS : LET ch=USR VAL "64330
": GO TO VAL "20"
 15 POKE VAL "23609",VAL "2": C
LEAR VAL "54783": LET hl=VAL "64
": LET d$="": LET h$="": INK VAL
 "5": PAPER VAL "0": CAT "xfer.b
in",: CAT "word.bin",: CLS : LET
 a=USR VAL "59081": RANDOMIZE US
R VAL "24322": GO TO VAL "10"
 20 CLS : POKE VAL "23658",VAL
"0": GO SUB VAL "8500": IF a=VAL
 "0" THEN GO TO VAL "3000"
 30 PRINT "                MENU"'
"p> print text"'"s> save text"'"
l> load text"'"y> read file"'"g>
 printer codes"'"t> save tasword
"'"b> bank select"'"x> file tran
sfer"'"d> change active drive"
 55 LET in=IN VAL "244"
 60 IF in=VAL "241" THEN LET b$
="DOCK": LET l$=d$: LET fl=dl
 65 IF in=VAL "1" THEN LET b$="
HOME": LET l$=h$: LET fl=hl
 70 LET drive=USR VAL "24358"
```

```
  80 PRINT AT VAL "12",VAL "0";"
Active Bank - "; INVERSE 1;b$; I
NVERSE 0;AT VAL "14",VAL "0";"Ac
tive Drive - "; PAPER VAL "2";CH
R$ drive
  85 PRINT AT VAL "16",VAL "0";"
le Name= ";l$;"                 ";"F
le length= ";: IF fl>64 THEN PR
INT fl;" bytes                  "
  87 IF fl<=VAL "64" THEN PRINT
AT VAL "17",VAL "13";"
                                "

  90 PRINT AT VAL "20",VAL "8";"
Press Any Key"
  95 LET a$=INKEY$: IF a$="" THE
N GO TO VAL "80"
 100 LET b=CODE a$: IF b<98 THEN
 LET b=b+VAL "32"
 110 IF b=VAL "100" THEN GO SUB
VAL "7500": GO TO VAL "80"
 115 IF b=VAL "98" AND in=VAL "1
" THEN OUT VAL "244",VAL "241":
GO TO VAL "55"
 135 IF b=VAL "98" THEN OUT VAL
"244",VAL "1": GO TO VAL "55"
 180 GO TO VAL "500"


 500>PRINT AT VAL "20",VAL "0";"
ENTER=cont., C=change choice"
 510 LET a$=INKEY$: IF a$="c" TH
EN GO TO VAL "20"
 520 IF CODE a$<>VAL "13" THEN G
O TO VAL "510"
 600 IF b=VAL "116" THEN GO TO V
AL "700"
 610 IF b=VAL "121" THEN CLS : G
O TO VAL "10"
 620 IF b=VAL "115" THEN CLS : G
O TO VAL "1000"
 635 IF b=VAL "120" THEN GO TO V
AL "8000"
 640 IF b=VAL "106" THEN GO TO V
AL "2000"
 650 IF b=VAL "112" THEN GO TO V
AL "200"
 660 IF b=VAL "103" THEN GO TO V
AL "300"
 699 CLS : GO TO VAL "20"
 700 RANDOMIZE USR VAL "24427"
 705 CLS : LET i=VAL "8": MOVE "
tasword.bas",15
 710 MOVE "xfer.bin",24320,143:
MOVE "word.bin",54784,10751: GO
TO VAL "1"
 950 POKE x,b-VAL "256"*INT (b/V
AL "256"): POKE (x+VAL "1"),INT
(b/VAL "256"): RETURN
1000 LET b=FN p(VAL "62216")
1005 CLS : PRINT AT VAL "8",VAL
"0";"Type File Name for Saving";
 PRINT '"Maximum 9 characters"'
""d"" to change drives"
1015 PRINT AT VAL "21",VAL "0";"
DRIVE= "; PAPER VAL "2";CHR$ dri
ve;AT VAL "21",VAL "10"; PAPER V
AL "8";"BANK= "; PAPER VAL "2";b
$
1020 INPUT LINE a$: IF a$="" THE
N : CAT "",: GO TO VAL "1015"
1025 IF a$="d" THEN GO SUB VAL "
7500": GO TO VAL "1015"
1035 IF in=VAL "1" THEN LET h$=a
$
1040 IF in=VAL "241" AND a$<>""
THEN LET a$=a$+"*": LET d$=a$
1050 IF LEN a$>VAL "10" THEN PRI
NT '"TITLE TOO LONG": PAUSE VAL
"200": GO TO VAL "1005"
1070 LET i=VAL "12": LET a$=a$+"
.bin"","+STR$ b+","+STR$ a: MOVE
 "a$",: CLS
1080 PRINT AT VAL "8",VAL "0";"t
ext file ";a$;" saved:";AT VAL "
10",VAL "0";a;" bytes,",INT (a/P
EEK VAL "62237");" lines"
1090 PAUSE 0: GO TO 20
2005 CLS : PRINT AT VAL "7",VAL
"0";"Type File Name";'''"     OR"'
'"Enter for Directory"''"""d"" t
o change drive"
2010 PRINT AT VAL "21",VAL "0";"
DRIVE= "; PAPER VAL "2";CHR$ dri
ve;AT VAL "21",VAL "10"; PAPER V
AL "8";"BANK= "; PAPER VAL "2";b
$
2025 INPUT LINE a$: IF a$="" THE
N CAT "",: GO TO VAL "2010"
2030 IF a$="d" THEN GO SUB VAL "
7500": GO TO VAL "2010"
2060 LET c=LEN a$: IF a$(c)="*"
AND in=VAL "1" THEN OUT VAL "244
",VAL "241"
2070 IF a$(c)<>"*" AND in=VAL "2
41" THEN OUT VAL "244",VAL "1"
2090 LET a=USR VAL "59081": LET
a=VAL "0"
2100 LET b=FN p(VAL "62216")
2115 IF a$(c)="*" THEN LET d$=a$
2120 IF a$(c)<>"*" THEN LET h$=a
$
2150 LET a$=a$+".bin"","": CAT "a
$",: GO TO VAL "10"


7000>DEF FN p(x)=PEEK x+VAL "256
"*PEEK (x+VAL "1")
7500 RANDOMIZE USR VAL "24388"
7510 LET drive=USR VAL "24358":
RETURN
8000 LET t1=VAL "19200": LET t=V
AL "33280": CLS : INPUT "Source
Bank (d/h) ";t$
8002 IF t$="h" THEN POKE VAL "24
320",VAL "1": POKE VAL "24321",V
AL "241"
8005 IF t$="d" THEN POKE VAL "24
320",VAL "241": POKE VAL "24321"
,VAL "1"
8030 INPUT "Starting Line ";s: L
ET s=t+(s*VAL "64"): GO SUB VAL
"8700": LET s1=s
8040 INPUT "Last Line ";s: LET s
=t+(s*VAL "64"): LET s=s-s1: LET
 s2=s: IF in=VAL "1" AND s2+dl>t
1 OR in=VAL "241" AND s2+hl>t1 T
HEN PRINT AT VAL "21",VAL "0";"T
OO LONG": PAUSE VAL "300": CLS :
 GO TO VAL "8040"
8045 GO SUB VAL "8710": GO TO VA
L "8600"
8050 INPUT "Insert at Line No.?
";s: LET s=t+(s*64): LET n=s: GO
 SUB VAL "8700"
8060 IF t$="d" THEN POKE VAL "24
320",VAL "1": LET s=(t+hl)-n: GO
 SUB VAL "8710": LET s=n+s2: GO
SUB VAL "8720": GO SUB VAL "8200
": POKE VAL "24320",VAL "241": L
ET s=s1: GO SUB VAL "8700": LET
s=s2: GO SUB VAL "8710": LET s=n
: GO SUB VAL "8720": GO SUB VAL
"8200": GO TO VAL "10"
8070 IF t$="h" THEN POKE VAL "24
320",VAL "241": LET s=(t+dl)-n:
GO SUB VAL "8710": LET s=n+s2: G
O SUB VAL "8720": GO SUB VAL "82
00": POKE VAL "24320",VAL "1": L
ET s=s1: GO SUB VAL "8700": LET
s=s2: GO SUB VAL "8710": LET s=n
: GO SUB VAL "8720": GO SUB VAL
"8200": GO TO VAL "10"
8200 RANDOMIZE USR VAL "24322":
RETURN
8505 LET a=ch: LET a=64*INT (a/6
4+VAL "0.99")
```

```
8520 IF IN VAL "244"=VAL "241" T
HEN LET dl=a
8530 IF IN VAL "244"=VAL "1" THE
N LET hl=a
8535 RETURN
8600 INPUT "Insert/Add (i/a) ";i
$: IF i$="i" THEN GO TO VAL "805
0"
8602 IF t$="d" THEN LET s=t+hl+V
AL "64": GO SUB VAL "8720": GO S
UB VAL "8200": GO TO VAL "10"
8605 IF t$="h" THEN LET s=t+dl+V
AL "64": GO SUB VAL "8720": GO S
UB VAL "8200": GO TO VAL "10"
8620 LET hi=INT (s/VAL "256"): L
ET lo=s-(hi*VAL "256"): RETURN
8700 GO SUB VAL "8620": POKE VAL
 "24326",lo: POKE VAL "24327",hi
  RETURN
8710 GO SUB VAL "8620": POKE VAL
 "24329",lo: POKE VAL "24330",hi
: RETURN
8720 GO SUB VAL "8620": POKE VAL
 "24323",lo: POKE VAL "24324",hi
: RETURN
```

Once all the changes have been made, GO TO 700. This will make a copy of the new Tasword to disc. Once you have become familiar with the transferring of files, I think you will find it useful.

Changing Tasword II screen colors:

As purchased, Tasword II uses a white border, white paper, and black ink. Even with an RGB monitor this combination can be a strain on the eyes. Here are some POKEs that will change the paper and ink colors to those of your own liking. These changes only effect the 64 column mode and not the 32. These changes, written by John Lambert, originally appeared in the May 1985 issue of Sinclair User magazine.

TEXT area	Border	Margins	Bottom Status lines
58512,54	64516,(0-9)	58508,54	64570,C
58513,C		58509,C	59993,C
58521,54			
58522,C			

The vaule of C is determined using this formula, (8 * paper color) + ink color. e.g. Red paper with Yellow ink, 2(Red) * 8 + 6(Yellow)=22. The border color is POKEd using a direct color number (0-9).

OS-64 & Spectrum on Disc

Using either OS-64 or a Spectrum emulator cartridge requires removing the disc interface. I need the RGB interface on the board to drive my monitor. The two routines listed below will relocate and run either operating system from the Dock bank. While you are stuck with tape again, you will have RGB.

With the computer turned off, disconnect the FD-68 interface board from the rear edge connector. Insert the cartridge and fire up the system. Next type in and RUN the following program. Use only one line 30 and one line 50.

```
 10 REM XXXXXXXXXXXXX (14 x's)
 20 CLEAR 32767
 30 FOR A=31515 TO 31528: REM OS-64
 30 FOR A=23760 TO 23773: REM Spectrum
 40 READ B: POKE A,B: NEXT A
 50 RANDOMIZE USR 31515: REM OS-64
 50 RANDOMIZE USR 23760: REM Spectrum
 60 DATA 243,33,0,0,17,0,128,1,
0,64,237,176,251,201
```

```
DI
LD HL,0000
LD DE,8000
LD BC,4000
LDIR
EI
RET
```

This routine will block move either operating system up to 32768. Now make a copy of this to tape, SAVE "OS-64" CODE 32768, 16384 or, SAVE "SPECTRUM" CODE 32768,16384.

Now that you have the operating system on tape, turn off the computer and remove the cartridge. Reinsert the FD-68 interface and fire up the computer again. CLEAR 32767. Now load the machine code back in from tape with LOAD ""CODE and then save a copy to disc with MOVE "OS-64.BIN",32768,16384 or, MOVE "SPECTRUM.BIN",32768,16384. To move the operating system down into the first 2 chunks of Dock bank RAM and initialize it, you will need the following routine.

```
 10 FOR A=24320 TO 24338
 20 READ B: POKE A,B: NEXT A
 30 DATA 243,62,3,211,244,33,0,
128,17,0,0,1,0,64,237,176,251,19
5,(5 for OS-64, 0 for Spectrum)
```

```
DI
LD A,03
OUT (F4),A
LD HL,8000
LD DE,0000
LD BC,4000
LDIR
EI
JP 0005 (0000 for Spec)
```

After you run the program, save a copy of the code generated from the basic with MOVE "CODE.BIN",24320,19 or, MOVE "SCODE.BIN",24320,19.

Delete lines 10,20, and 30, and type in the following line. Use only one of the following lines.

```
 10 CAT "OS-64.BIN",: CAT "CODE    For OS-64
.BIN",: RANDOMIZE USR 24320
 10 CAT "SPECTRUM.BIN",: CAT "S    For Spectrum
CODE.BIN",: RANDOMIZE USR 24320
```

Save this basic to disc as MOVE "OS-64.BAS",1 or MOVE "SPECTRUM.BAS",1. To use either system, CAT "OS-64.BAS", or CAT "SPECTRUM.BAS",. The Only drawback, is that once you initialize the operating system, you will again be tied to cassette tape for programs.

XFER BYTES

```
ADDR HEXCODE  MNEMONIC          NOTES
5F00 01       1                 ;Source Bank Chunks to be enabled
5F01 F1       241               ;Destination Bank Chunks to be enabled
5F02 110082   LD DE,8200        ;Load DE with dest. address
5F05 210082   LD HL,8200        ;Load HL with source address
5F08 01007E   LD BC,7E00        ;Load BC with # of bytes to transfer
5F0B 3A5B68   LD A,(5F00)       ;Load Accumulator with Source chunks
5F0E D3F4     OUT (F4),A        ;Enable source chunks
5F10 7E       LD A,(HL)         ;Load Accumulator with contents of
                                ; source address-value being transfered
5F11 08       EX AF,AF'         ;Store Accumulator contents in
                                ; alternate Accumulator
5F12 3A5C68   LD A,(5F01)       ;Load Accumulator with Dest. chunks
5F15 D3F4     OUT (F4),A        ;Enable destination chunks
5F17 08       EX AF,AF'         ;Load Accumulator with the contents of
                                ; its alternate- value being transfered
5F18 12       LD (DE),A         ;Load dest. address with value of Acc.
5F19 13       INC DE            ;Increment Destination Address
5F1A 23       INC HL            ;Increment Source Address
5F1B 0B       DEC BC            ;Decrement # of bytes to be transfered
5F1C 78       LD A,B            ;If # of bytes left
5F1D B1       OR C              ; to transfer does not equal
5F1E 20EB     JR NZ,5F0B        ; zero then jump to 5F0B
5F20 3A5C68   LD A,(5F01)       ;If # of bytes left to transfer
5F23 D3F4     OUT (F4),A        ; equals zero then enable destination
5F25 C9       RET               ; chunks and RETurn
```

SELECT DRIVE

```
ADDR HEXCODE  MNEMONIC          NOTES
5F44 DBF4     IN A,(F4)         ;Fetch and store currently selected
5F46 08       EX AF,AF'         ; chunks for later return
5F47 3E03     LD A,03           ;Enable chunks 0 & 1- Dock Bank
5F49 D3F4     OUT (F4),A        ; OUT 244,3
5F4B 3AED3F   LD A,(3FED)       ;Load Accumulator with (curdrv)
5F4E 1E88     LD E,88           ;Test to see if curdrv="D"
5F50 BB       CP E              ; if it does then
5F51 3015     JR NC,5F67        ; jump to 5F67

5F53 E60F     AND 0F            ;If curdrv didn't equal "D" then mask
                                ; high nybble and
5F55 CB27     SLA A             ;Increment curdrv

5F57 4F       LD C,A            ;Store low nybble in Register C
5F58 3AED3F   LD A,(3FED)       ;Load Accumulator with (curdrv)
5F5B E6F0     AND F0            ;Mask low nybble
5F5D 47       LD B,A            ;Store high nybble in Register B
5F5E 79       LD A,C            ; bring back low nybble to Accum.
5F5F B0       OR B              ;Reasseable low nybble & high nybble
5F60 32ED3F   LD (3FED),A       ;Store update in (curdrv)
5F63 08       EX AF,AF'         ;Bring back previously stored chunks
5F64 D3F4     OUT (F4),A        ; and enable them
5F66 C9       RET               ;RETurn to basic

5F67 3E01     LD A,01           ;If curdrv ="D" then reset  curdrv to
5F69 18EB     JR 5F57           ; "A" and jump back to 5F57
```

READ DRIVE

```
ADDR HEXCODE  MNEMONIC          NOTES
5F26  DBF4    IN A,(F4)         ;Fetch and store currently selected
5F28  08      EX AF,AF'         ; chunks for later return
5F29  3E03    LD A,03           ;Enable chunks 0 & 1- Dock Bank
5F2B  D3F4    OUT (F4),A        ; OUT 244,3
5F2D  3AED3F  LD A,(3FED)       ;Load Accumulator with (curdrv)
5F30  E60F    AND 0F            ;Mask high nybble of (curdrv)
5F32  5F      LD E,A            ;Load E register with the contents of A
5F33  3E08    LD A,08           ;Load the Accumulator with 8
5F35  014400  LD BC,0044        ;Load BC register with ASCII code "D"
5F38  B8      CP E              ;Compare E register with Accumulator
5F39  2805    JR Z,5F40         ; if result=0 (a match) then exit
5F3B  CB2F    SRA A             ;If no match, then divide Acc. by 2
5F3D  0B      DEC BC            ; and decrement ASCII code by 1 and
5F3E  18F8    JR 5F38           ; go back and compare again with curdrv
5F40  08      EX AF,AF'         ;Bring back previously stored chunks
5F41  D3F4    OUT (F4),A        ; and enable them
5F43  C9      RET               ;RETurn to basic
```

RESET CODE

```
ADDR HEXCODE  MNEMONIC          NOTES
5F6B  3E01    LD A,01           ;  This routine is only called when
5F6D  D3F4    OUT (F4),A        ;  you go to make a back up copy of
5F6F  32005F  LD (5F00),A       ;  Tasword with the "t" option.
5F72  3EF1    LD A,F1           ;  It restores the source and dest.
5F74  32015F  LD (5F01),A       ;  chunks, the source and destination
5F77  AF      XOR A             ;  addresses, and the number of bytes
5F78  32035F  LD (5F03),A       ;  to be transfered for initial load in
5F7B  32065F  LD (5F06),A       ;  from disc. Without it, you may be
5F7E  32095F  LD (5F09),A       ;  saving code that contains data
5F81  3E82    LD A,82           ;  regarding the last XFER of files you
5F83  32045F  LD (5F04),A       ;  made, resulting in a nice crash.
5F86  32075F  LD (5F07),A       ;  This routine may be eliminated when
5F89  3E7E    LD A,7E           ;  using the XFER BYTES code with
5F8B  320A5F  LD (5F0A),A       ;  another program.
5F8E  C9      RET               ;
```

Building Blocks

All the projects I work on start out as little bits of code or Basic. I build a program a little at a time, taking time to be sure that each routine works before I go on to the next. After I have a number of these "blocks" completed, I tie them all together with a "handler" with a menu that will call the various routines. I hope to be able to present various routines or "Building Blocks" each issue which can be used together or alone depending on the application you have in mind.

Applying XFER BYTES

The XFER BYTES code can be used to transfer data between banks or within the same bank. To use XFER BYTES in your own programs for data transfer, you will need to supply the source and destination chunks, the source and destination addresses, and the number of bytes to be transfered. If you are using the code in its original memory location (24320), you can use the Basic lines 8620,8700,8710,and 8720 from Tasbank II. All that will be left for you to do is to POKE 24320 with the source enable chunks, and 24321 with the destination enable chunks. To move data within the same bank, POKE the source and destination chunks (24320 & 24321 respectively) with the same number. After all that is done, RANDOMIZE USR 24322 to activate.

The XFER BYTES code as written, resides starting at memory location 24320. At this address it is out of the way and doesn't use any RAM. However, if you are using the print driver software from John Oliger, you will notice that his screen copy routine starts at this same memory address. Therefore, the XFER BYTES code can be relocated with little fuss.

Let's say you wish to relocate the code at address 65000. LET base=65000. Load the code in from the disc with an offset with CAT "xfer.bin",base.

```
POKE base, source chunks (0-255)
POKE base + 1, destination chunks (0-255)      These POKEs will
POKE base + 3, lo byte of destination          need to be made
POKE base + 4, hi byte of destination          from within your
POKE base + 6, lo byte of source               programs.
POKE base + 7, hi byte of source               Refer to the m/c
POKE base + 9, lo byte of length of transfer   notes on page ??.
POKE base +10, hi byte of length of transfer
```

The following POKEs will only be necessary once at the time of relocation. They are the addresses within the code that reference the source and destination chunk enable information.

```
POKE base +12, lo byte of base --- source chunk enable info
POKE base +13, hi byte of base      "     "     "     "
POKE base +19, lo byte of base +1- destination chunk enable info
POKE base +20, hi byte of base +1     "       "     "     "
```

Activate with RANDOMIZE USR base+2 (e.g. RANDOMIZE USR 65002)

IMPORTANT! When using XFER BYTES, NEVER disable the chunk of memory in which the code resides. If the code is in chunk 5 of Home bank, DO NOT enable chunk 5 of the Dock bank. Enabling chunk 5 Dock will DISable chunk 5 Home and the computer will "lose sight" of the code and crash.

READ DRIVE & SELECT DRIVE

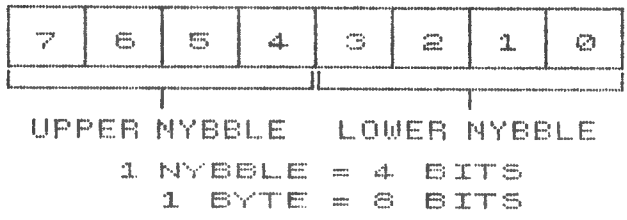
The initial version of this routine was sent to be by Jack Dohany. He admitted there were a few bugs in it. As a reader stated in a letter, when he finds bugs in his programs, he gets out a can of "RAID" and goes after the little devils. With the aid of Dave McNeely again I was able sort out the bugs. Once it worked, I added a few things to make it easier to use, with a minimum of Basic to control it. The Read and Select routines are completely relocatable to any address you desire. As written READ drive starts at address 24350, and SELECT drive starts at 24388. To relocate the routine you should use the same format as for XFER BYTES. Let your new starting address equal "base". Let's use 65000 again.

```
LET base=65000.
Read Drive = base
Select Drive = base + 38
```

To Read the current drive, use LET drive=USR base. When the routine is done and it returns to Basic, the variable "drive" will contain the ASCII code for the letter A,B,C,or D. Then simply PRINT CHR$ drive to obtain the current drive. To use Select drive, RANDOMIZE USR base + 38. This will increment the current drive by one. If current drive enabled was "A" then it would change to "B", "B" to "C", "C" to "D", or "D" back to "A".

In the notes on the machine code, I use the term "nybble". A "byte" is equal to 8 bits. A "nybble" is 4 bits or half a "byte". You have an upper nybble and a lower nybble. Refer to the drawing in Figure 3.

FIGURE 3



BLOCK MOVE (LDIR)

In the first newsletter I mentioned that I used the Dock bank to store screen images. I transfered them in and out with the block move (LDIR) routine. A reader stated that I mentioned the block move rather casually as if you all had one. Well, the fact is, all of you, or for that matter, anyone with a Z80 based computer already has this routine. The routine I refer to is the Z80 machine code instruction LDIR. This instruction will move a specified block of memory from one address to another.

The LDIR instruction requires three pieces of information to operate. Before calling the LDIR, first it needs to know the source address, the destination address, and the number of bytes to be moved. The HL register pair must be loaded with the source address, the DE register pair with the destination address, and the BC register pair with the number of bytes to be moved. Then the LDIR instruction can be called. In the blink of an eye, the task will be completed.

In the use of machine code, you will always see and hear of

the term "Registers". A register can be thought of as a memory address not in RAM, but inside the CPU. Actually it is more of a place rather than an actual address (if you know what I mean). Data can be stored and manipulated within these registers just as you would memory addresses in RAM. To put information in a register you must LD (load) it. This would be the equivelant of POKEing a memory address from Basic. Unlike RAM addresses, Registers can only be accessed, or loaded from within machine code. They are not accesible from Basic.

The following Basic program was written by John Kuhn of THE SINC TIMES.

```
   1 REM xxxxxxxxxxxx (12 x's)
   2 POKE 26715,33: POKE 26718,1
7: POKE 26721,1: POKE 26724,237:
 POKE 26725,176: POKE 26726,201
   3 INPUT " MOVE FROM ADDRESS "
;ST: INPUT " MOVE TO ADDRESS ";F
N: INPUT " # OF BYTES ";NO
   4 POKE 26716,ST-256*INT (ST/2
56): POKE 26717,INT (ST/256): PO
KE 26719,FN-256*INT (FN/256): PO
KE 26720,INT (FN/256): POKE 2672
2,NO-256*INT (NO/256): POKE 2672
3,INT (NO/256): PAUSE 0: RANDOMI
ZE USR 26715: PAUSE 0
```

If you would like to store screen images in the Dock bank and then call them up from within your programs:

1) OUT 244,241
2) CAT "name.scr",
3) GOTO 1
4) Enter source as 16384 (start of the display file)
5) Enter destination as 32768
6) Enter length - 6912= color screen (pixels + attributes)
   6144= black and white screen (pixels only)
   The Dock bank will hold 4 color screens, or 5 B&W.
7) Press any key to block move it to the new location

To store more than one screen, add the number of bytes that correspond to the type of screen you are moving. For color screens, add 32768+6912=39680; 39681 is the destination for the next color screen. For black and white screens add 32768+6144= 38912; 38913 is the destination for the next black and white screen. Use the same technique for additional screens.

To call a screen back into the display file, go to step 4 and reverse the source and destination addresses. To call up a screen image from the Dock bank from within your programs, first OUT 244,241, then use an abbreviated version of the above Basic, eliminating the PAUSE statements in line 4, the INPUT statements of line 3, and define the 3 variables ST, FN, and NO, prior to entering line 4. Immediately after calling line 4, do an OUT 244,1 to return control to your program.

BOOT

Here is a boot program that I use on all my discs. Loading programs is as simple as touching a few keys. No more typing errors to worry about. I think you'll like it. Type it in and have at it.

```
   1 REM Disk Auto-boot Loader
     Written by Chia-Chi Chao
  10 BORDER 0: PAPER 0: INK 7: C
LS : OVER 1: LET x=0: LET y=5: L
ET z=0: LET a=x: LET b=y: LET c=
z
  20 OUT 244,1: CAT "",: PRINT #
1; OVER 0; PAPER 2;"5-8 Select F
ile     9-0 Command"
  30 PRINT AT y,x; INVERSE 1;"
                ";  INVERSE 0;AT 21
,0; OVER 0; "Run     Load     Cat
alog Basic": PRINT AT 21,z; INVE
RSE 1;"                "
  40 LET a$=INKEY$: IF a$="" THE
N GO TO 40
  50 IF a$="8" AND x=0 THEN LET
x=16
  60 IF a$="5" AND x=16 THEN LET
 x=0
  70 IF a$="6" AND y<20 THEN LET
 y=y+1
  80 IF a$="7" AND y>5 THEN LET
y=y-1
  90 IF a<>x OR b<>y THEN PRINT
AT y,x; INVERSE 1;"
    ";AT b,a;"                ":
LET a=x: LET b=y: GO TO 40
 100 IF a$="0" AND z<24 THEN LET
 z=z+8
 110 IF a$="9" AND z>0 THEN LET
z=z-8
 120 IF z<>c THEN PRINT AT 21,z;
 INVERSE 1;"                ";AT 21,c;"
        ": LET c=z: GO TO 40
 130 IF CODE a$<>13 THEN GO TO 4
0
 140 IF z=16 THEN RUN
 145 IF z=24 THEN OVER 0: STOP
 150 LET a$=""
 160 FOR l=x+14 TO x+5 STEP -1:
IF SCREEN$ (y,l)<>" " THEN GO TO
 180
 170 NEXT l: BEEP .3,24: GO TO 4
0
 180 OVER 0: FOR m=x+1 TO l: LET
 a$=a$+SCREEN$ (y,m): NEXT m
 190 IF a$(LEN a$-2 TO )="BAS" A
ND z=8 THEN LET a$=a$+",10000"
 200 IF z<=8 THEN CAT "a$",
9998 STOP
9999 MOVE "BOOT.BAS",10
```

As I'm sure most of you know, the RP/M operating system is now available from Aerco. RP/M is an enhancement of CP/M 2.2. This enables us to run CP/M 2.2 software. There is an enormous amount of public domain CP/M 2.2 software out there.

If you join one of the hundreds of CP/M user groups, and request program material, you will be asked what format you need the disc configured for, Osborne, Morrow, Kaypro, IBM, Apple, Commodore, etc. Timex is not one of your choices. The programs are the same, but they are stored on the disc in different formats. Jerry Chaskis at Aerco now has the RP/M emulating the Morrow MD3 computer. The RP/M boot disc now available, has the capability of reading CP/M discs in the Morrow format.

Listed below are a few sources of CP/M software.

FOG
P.O. Box 3474
Daly City, CA 94015-0474
(415) 755-2000

CP/M User Group
2248 Broadway
New York, NY 10024

SIG/M
Box 97
Iselin, NJ 08830

Micropro
1299 4th St.
San Rafael, CA 94901

I am told that if you write to the FOG group, you will receive a sample newsletter 65 pages long, which also contains a list of available programs. They also have discs in the Morrow format. When you write to any one of the CP/M groups, ask about the availability of discs in the Morrow format. I know of people that have received discs in the Morrow format and they load and operate fine.

There have been frequent updates and improvements to the RP/M. Jerry says that the operating system itself is now quite stable, with only an occasional tweak. Most of the current updates have been to programs contained on the boot disc. The MODEM750 program is for using the 2050 modem with the RP/M. MODEM750 is still a little rough on the edges, but it does work both uploading and downloading.

Aerco is also offering a reconditioned, ADDS Viewpoint Terminal for $129. To use the terminal you will need an RS232 interface. Aerco has a dual RS232 interface available for $99. You can buy the terminal and the RS232 as a package for $199. I purchased the RP/M, the terminal, and the RS232. The terminal looks as though it is new. I was impressed with this "reconditioned" terminal. The green screen is clear and very easy to read. The detactched keyboard has 72 keys which include a numeric keypad. It has nice response and a general good feel to it. The keys on the keyboard show no visible signs of wear.

Two programs on the RP/M boot disc, TMXON, and TMXOFF, toggle, or enable and disable the 2068 screen and keyboard. The Timex writes to the screen at a rate of about 1200 baud. The terminal can write to the screen at 9600 baud. Once the terminal is no longer slowed by the 2068 screen, the true speed of the terminal can be appreciated. Printing to the screen is 8 times faster.

RP/M is a disc intensive operating system. This means frequent access of the disc.

One plus in opting for the terminal package is that when not being used for RP/M, the dual RS232 interface is at your command. There are driver programs for the RS232 on the Aerco boot disc, enabling the use of other modems, a serial printer, or a variety of other RS232 controlled devices.

The 80 column display is very readable using an RGB, monochrome, or composite monitor. Even a black and white TV is acceptable, but a color TV is the absolute pits. The 2068 display while readable, is no match for the terminal. The difference is night and day. It's honestly that much clearer!

The Aerco system formats the disc in 5K chunks. If a program is 1-4999 bytes long, it will use 5K of disc space. (e.g. a 1K program will use 5K on the disc etc.) The RP/M system formats in 2K chunks, which in turn makes for more efficient use of available disc space. Instead of the usual 31 program titles available with the Aerco system, the RP/M allows up to 126 titles per disc.

Word processing with the RP/M is great. Being able to see 80 columns on the screen is a real plus. Most word processors for the Timex system allow printing more than the 64 columns displayed, but viewing them requires moving a 64 column window across the text (a pain). With RP/M, what you see is what you get.

To realize the full potential of this operating system and make the best use of it, will require some homework. It is recommended that you pick up a book on CP/M. One such book is "CP/M Revealed," published by Hayden.

For those of you interested, Heath/Zenith, makers of Heathkit electronic kits and computers, offers a self-instructional course in CP/M. The catalog describes it as, "A beginner-oriented course that requires no previous background. Starting at ground level, this audio-tutorial based course first shows you how to use the most basic commands. Use of the resident text editor is also covered. This course is designed for use with computers having at least 48K of RAM and able to use the CP/M operating system" (that's us). Included is a large manual.

Upon successful completion of the course, you can receive college credits. As anyone familiar with Heath products knows, when it comes to electronics, either kits or instructions, there is no equal. Many schools use their courses in the classroom.

For more information and a catalog, write to:

Veritechnology Electronics Corporation
P.O. Box 167
St. Joseph, MI 49085

I would like to start a column dedicated to CP/M. I am VERY new to CP/M. My only experience with computers has been with Sir Clive's machines. If someone out there would like to author this column please drop me a line. I foresee a lot of good things for the RP/M in the near and distant future.

```
DIR
Name    Ext Bytes   Name    Ext Bytes   Name    Ext Bytes   Name    Ext Bytes
1200BAUDCOM    2K ! 300BAUD COM    2K ! 9600BAUDCOM    2K ! ASCII   COM    8K
B/W     COM    2K ! CALL    COM    2K ! CATALOG COM    2K ! CATALOG DOC    2K
CATALOG TMP    2K ! COLOR   COM    2K ! D       COM    4K ! DEL     COM   18K
DIFFB   COM    2K ! DIR     COM    4K ! DO      COM    2K ! DUMP    COM    2K
ENARCH  COM   10K ! FILES   DOC    4K ! FINDBAD COM    2K ! FLAWTBL COM    2K
FLAWTBL DOC    2K ! FORMAT68COM    2K ! GETSYS  COM    2K ! GREP    COM   12K
GREP    DOC    6K ! LOCKOUT COM    2K ! MEMMAP  COM    2K ! MODEM7  DOC   12K
MODEM750COM   10K ! PUTSYS  COM    2K ! REPOS   COM    2K ! RESTOR  COM    2K
RESTOR  DOC    2K ! RPMPIP  COM    8K ! RPMPIP  DOC   12K ! RSTAT   COM    6K
RSTAT   DOC    2K ! SCRAMBLECOM    2K ! SCRAMBLEDOC    2K ! SHOWDISCCOM    2K
SUBREN  COM    2K ! SUBREN  DOC    4K ! TED     COM   18K ! TED     DOC   24K
TMXOFF  COM    2K ! TMXON   COM    2K ! TR      COM    4K ! TRANSLATCOM    4K
UNARCH  COM   10K ! USQ     COM    2K ! XSUB    COM    2K ! Z8E     COM   12K
Z8E     DOC  102K ! ZASSEM  COM   10K ! ZASSEM  DOC    4K
55 File(s), occupying 370K of 376K total capacity
70 directory entries and 6K bytes remain on A:
SHOWDISC


Drive A:
Each block:      2 K bytes
Disc size:     380 K bytes

Allocation vectors:

   11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
   11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
   11111111 11111111 11111111 11111111 11111111 11111111 11111111 111000

a0>MEMMAP

------------------------------------------------
System: RP/M equivalent to CP/M 2.2
0000 "warmboot" vector
0005 RDOS vector
005C default file control block (FCB)
0080 RP/M record buffer
0100 first word address (fwa) of user area
D3FF last word address (lwa) of user area
D400 RCP fwa
DC00 RDOS fwa
DC06 RDOS entry point
EA00 CBIOS fwa
EA03 CBIOS warmboot entry point

------------------------------------------------
RSTAT
A: R/W, Space:     6k

a0>FINDBAD

FINDBAD - ver 5.4
Bad sector lockout program
Universal version

Type CTL-C to abort

Testing directory area...

Testing data area...
**************************************************************************

        No bad blocks found
```

Loader IV and Loader V

by Syd Wyncoop

I have heard from a friend, Rod Gowen, that there is a lot of chatter on Compuserve about Kurt Casby's Loader IV and Loader V not working with the AERCO disc drives. I have mine working fine and following are the neccesary changes and their line numbers. Hope some other subscribers can make use of it.

Changes to Loader V

```
 1 ON ERR GO TO SGN PI: CLEAR
VAL "29863": LET bs=VAL "244": L
ET p=VAL "24007": LET f=VAL "20"
: LET a=VAL "119": LET o=NOT PI:
 OUT bs,o: LET i=SGN PI: LET b=i
+i: LET n$=" LOADER V - @1985  b
y K.A.Casby ": BORDER i: PAPER i
: INK VAL "9": OUT a,o: CLS : PR
INT n$''''''TAB PI+PI;"MAIN MENU
:"''TAB PI+PI;"1) RUN MTERM"''T
AB PI+PI;"2) DIALING MENU"''TAB
PI+PI;"3) LOAD BUFFER": GO SUB P
I: IF LEN z$<>i OR z$<"1" OR z$>
"3" THEN RUN
```

Be sure to type exactly as appears to avoid any errors. Errors will return you to the menu for no apparent reason!

```
 60 OUT bs,i: CAT "b:",: INPUT
"Filename? "; LINE y$: CLS : PRI
NT n$'''"Name of file: ";y$''''"C
arriage returns:"''"1) None"''"2)
 Each line"''"3) Each paragraph":
 GO SUB PI: IF LEN z$<>i OR z$<"
1" OR z$>"3" THEN RUN
```

Reset bank for Aerco disc drives and switch to drive B. Catalog will prevent misspell errors on name. Cannot overwrite disc A.

```
 61 PRINT AT PI+PI,VAL "18";z$'
''''''"Insert disc with ";y$;" f
ile"''''"& press key": PAUSE o:
LET y$=y$+".bin"","+STR$ VAL "29
864": RANDOMIZE USR VAL "23788":
 CAT "y$",: POKE p,VAL z$: PRINT
 : PRINT FLASH i;" LOAD complete
, press any key!  ": PAUSE o: RU
N
```

Pause to be sure disc is in place. Load file and run Loader V to reset banks to Home Rom.

```
9000 CAT "Ldr5.bin",: CAT "MTERM
.bin",: RUN
```

Autostart here with a MOVE "Ldr5.bin",9000. Delete all lines after 9000 before saving.

Changes to Reloader

```
 1 ON ERR GO TO SGN PI: CLEAR
VAL "29863": LET bs=VAL "244": L
ET f=VAL "5": LET o=SIN PI: OUT
bs,o: LET i=SGN PI: LET b=i+i: L
ET c=VAL "23788": LET n$=" Re-Lo
ader  @1985  by K.A.Casby ": BOR
DER i: PAPER i: INK VAL "7": CLS
 : PRINT n$''''''''TAB VAL "7";"1
) RUN MTERM"''''TAB VAL "7";"2)
LOAD BUFFER": GO SUB PI: IF LEN
z$<>i OR z$<"1" OR z$>"2" THEN R
UN
```

Explanations are same as those for Loader V, except autostart at line 1.

```
 10 OUT bs,i: CAT "b:",: INPUT
"Filename? "; LINE y$: CLS : PRI
NT n$'''"Name of file: ";y$''''"C
arriage returns:"''"1) None"''"2)
 Each line"''"3) Each paragraph":
 GO SUB PI: IF LEN z$<>i OR z$<"
1" OR z$>"3" THEN RUN
 11 PRINT AT PI+PI,VAL "18";z$'
''''''"Insert disc with ";y$;" f
ile"''''"& press key": PAUSE o:
LET y$=y$+".bin"","+STR$ VAL "29
864": RANDOMIZE USR c: CAT "y$",
: POKE VAL "24007",VAL z$: PRINT
 : PRINT FLASH i;" LOAD complete
, press any key!  ": PAUSE o: GO
TO f
```

Dockside

Barry A. Eisen

First of all, let me give my congradulations to David. A quarterly newsletter is no small feat. I hope that this becomes the definitive magazine for FD-68 users.

Getting down to business.

The first newsletter stated that the corrected exROM from Ray Kingsley was found to be incompatable with the FD-68 disc system. The FD-68 interface actually is compatable with the Ray Kingsley exROM. The problem, is that the exROM fixes that Kingsley used, were not those stated in the TIMEX technical manual. AERCo, on the other hand, did use the TIMEX fixes. It is possible to use both at once. With the original exROM and interface (with drives) on line, MOVE to disc,the entire initial boot memory as "boot.bin",16384,49150. Make back-ups. Now, all that you have to do if you want to run programs that use these fixes, is boot up "boot.bin", and all problems should be solved. After testing this program by booting it up and checking it, you are ready to install the Kingsley rom. To insure compatability, the command "OUT 244,1" must be directly in front of ALL disk commands. CAT "", would need to be rewritten OUT 244,1: CAT "",. Even one other command between the two will cause an error code and halt the program. They must be in the same line also. A rewrite of the AERCo EPROM would free up enough space for some of the other promised commands.

One thing that disturbs me is that we won't be able to exchange disks with people who have other disk systems. In a future article I hope to explain some quirks of the computer itself. These are important once you realize just how easy it is to switch between games now that you are freed from the time and bother of tapes. Compilers and assemblers have to be modified to use the disks. Programs such as Malcolm Evens' MACHINE CODE TUTOR program has to have its headers modified. just like MTERM II's SAVE code. Again hopefully, a subject for a future article.

Till next time, just keep hacking

Chunk Enable Update

In the first issue I stated that enabling any of the lower 4 chunks of the Dock bank would result in a crash. This was an obvious error that slipped through undetected until after the newsletter went to press. It should have stated that only the first chunk (chunk 0) of the Dock bank should be directly enabled (OUT 244,1). Directly enabled? Read on.

When the computer is fired up, the contents of the EPROM are block moved into the 2nd chunk (chunk 1) of the Dock bank. Then a copy of the 1st 8K of the Home ROM (chunk 0) is copied into the 1st chunk of the Dock bank RAM (chunk 0) where the "bugs" in its design can be repaired. Next, the fixes, plus a few alterations are to made to the copy of the Timex ROM in the Dock bank RAM chunk 0.

Timex never fully implemented the code to properly interpret the keywords CAT, MOVE, ERASE, and FORMAT. When the Home ROM (OUT 244,0) is presented with a disc command, the J Invalid I/O device 0:1 error will appear. If you OUT 244,1 you will be reading the altered ROM as it appears in chunk 0 of the Dock bank. Now, when this altered ROM is presented with an error of ANY type, it will jump to chunk 1 of the Dock bank. At this point the error will be tested to see if it is a J Invalid I/O device. If it is, a jump will be made to a table of allowable disc commands. If the command which caused the error is in fact a legal disc command it will be executed. If there is a disc error, the appropriate message will appear on the screen. If the error is not disc related (not a J) it will return to the error handler in Home bank and proceed accordingly.

Think of the Aerco DOS as an extension, or addition to the existing operating system, not as a replacement of it. In addition to the disc commands, chunk 1 is also used as a buffer to store the current drive (curdrv), the directory as it appeared at the last CAT "", command, as well as a track buffer to name but a few. Therefore, accessing chunk 1 of the Dock bank should only be done from within a machine code program which would enable it, read it, and then disable it again before returning. Chunk 1 of the Dock bank is available only by referral. You must be referred to this area by machine code, either directly or indirectly. A direct access would be one which is made from within a machine code program (refer to the article on READ & SELECT drive pg. 8). Indirect access would be in the form of a CAT "", command which would in turn call chunk 1 and search for a match to the command. If you just walk in the front door of chunk 1 without a referral, you will promptly have the door slammed in your face and be locked out. In most cases you will need to turn off the computer to regain control.

Once the computer is fired up, only the first 2 chunks (0&1) of the Dock bank contain any information. Enabling chunks 2&3 would be like opening a book and finding nothing but blank pages. In the case of chunks 2&3, when you enable them in preference over the same chunks in the Home bank, the computer expects to find something there such as the system variables, the stack, etc., not just lots of empty space.

The one exception to this would be if you had a Language oriented program in the Dock bank, such as the Spectrum operating system. If a cartridge is plugged in at the same time as the disc interface, the disc will take priority over whatever is plugged into the cartridge port. This renders helpless, any Spectrum emulator cartridge, or OS-64 for example. Doing an OUT 244,3 under these circumstances will only result in a crash and will NOT access the cartridge.

REVIEW

Disk File Manager
Chia-Chi Chao
73 Sullivan Drive
Moraga, CA 94556

Aerco Disk File Utility
Price: $16.00 ppd. Send SASE for information.

After Disk File Manager is booted in, it draws a 5-1/4" diskette on the screen which is followed by the command menu. Below is a brief description of the command structure used in the program.

1. Regular Catalog - Same as a CAT "", command.

2. Detailed Catalog - Does a standard header of the disc, supplying the auto-run line for Basic, and the starting address and length for machine code. It also displays the values written to port 255 (selects the video mode) and port 244 (horizontal select register- bank switching port). There is also an option which will display the disc track(s) each program occupies.

3. Check Disc - Checks for conflicts in disc tracks. Gives 4 different error reports:
   Track n taken more than once
   Track n is not marked on disk
   Track n is not available
   Track n is preoccupied

4. Copy Files - Allows the copying all files or just selected ones. Will copy to the same disc or a different one, making it easier for single drive users.

5. Display Occupied Tracks - Displays all disc tracks. Occupied tracks are printed in inverse.

6. Change Default Drive - Change the drive that will be accessed if no other drive is specified.

7. Printer Options - Outputs to the 2040 printer or to an 80 column printer. You will need to load your own print driver.

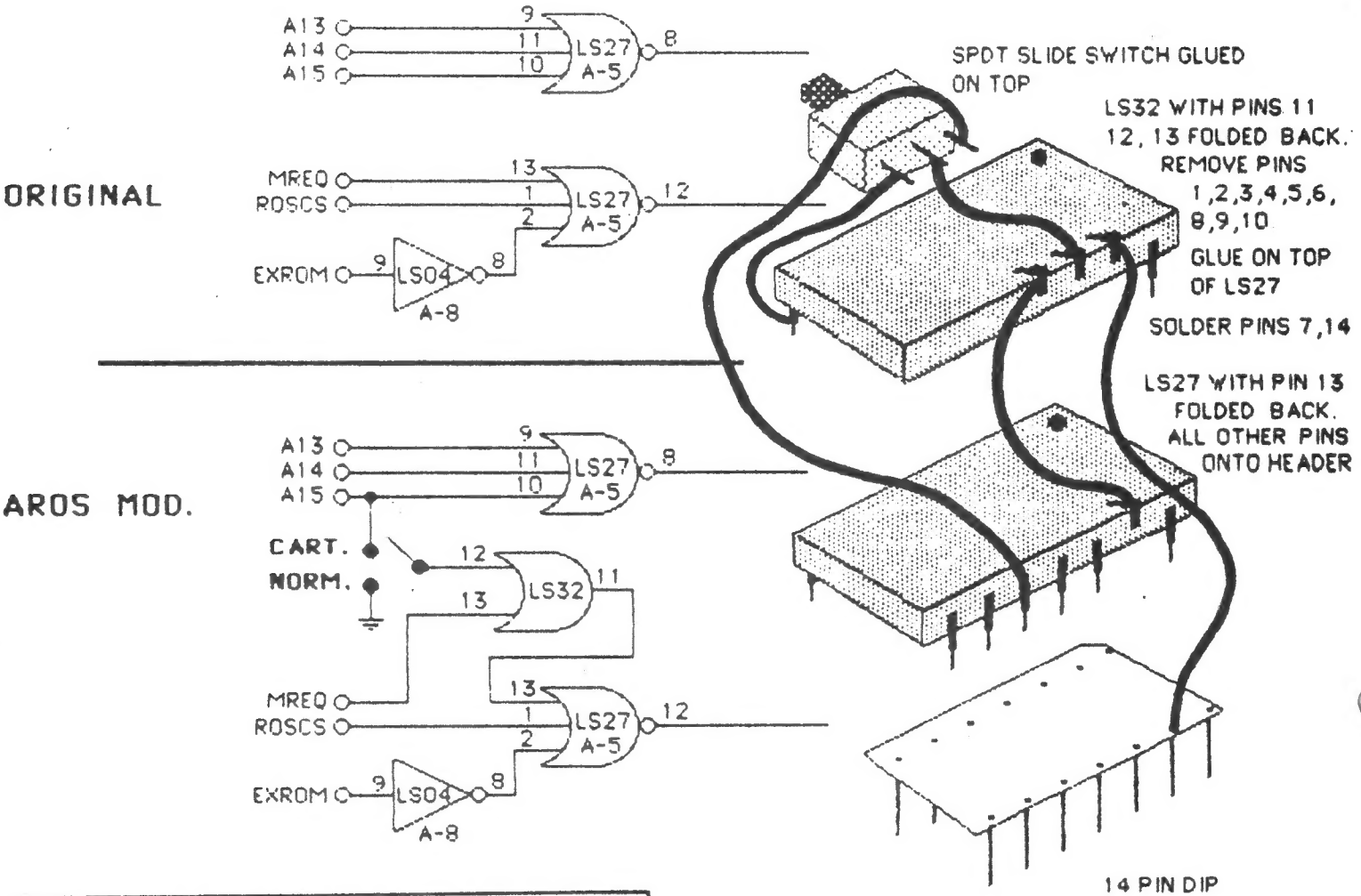
8. Exit to Basic - Does a NEW. Resets the computer without switching it off and on.

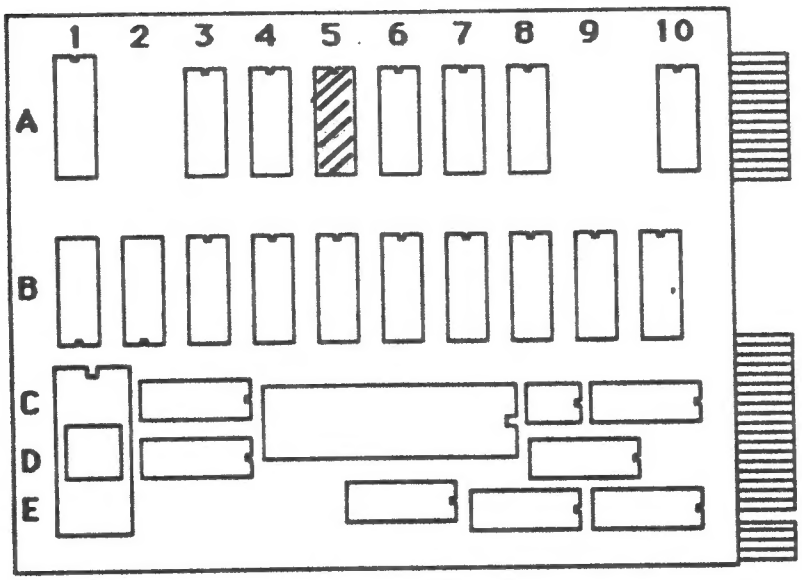
I find the Disk File Manager a handy utility. It provides useful information, is user friendly, and easy to use. The Check Disk instructions contain a demonstration of what happens when you save programs with the same name more than once.

One tip from the text: "To avoid track conflicts, always ERASE a file instead of overwriting it with the same filename, unless the new file has the same length or is longer than the old one."

Reviewed by Dave Hill

MOVING AN AROS CARTRIDGE ONTO YOUR AERCO DISK SYSTEM





AERCO
ACME ELECTRIC ROBOT CO.
Box 18093 Austin, . TX 78760
(512) 451-5874

REMOVE THE LS27 FROM THE SOCKET AT LOCATION A5 ON THE FD68 BOARD AND REPLACE WITH THIS ASSEMBLY. SET SWITCH TO CARTRIDGE AND MOVE "NAME.ARO", SET SWITCH TO NORM AND CAT "NAME.ARO",

YOU CAN BUY THIS ASSEMBLY FROM AERCO FOR $15 OR SEND YOUR CARTRIDGE AND $10 TO AERCO TO HAVE IT PUT ON DISK (PRICES INCLUDE RETURN SHIPPING).

Program and Article Authors

If you have a question about an article in this newsletter, write to the author. Who better to answer your questions?

Chia-Chi Chao 73 Sullivan Dr. Moraga, CA 94556	Leland W. Fiscus C/O USPS/MTSC 111 Chesapeake Norman, OK 73069	Syd Wyncoop 2107 SE 155th Portland, OR 97233
Barry Eisen 1415 Browning Rd. Pittsburg, PA 19141	C.A.T.S. P.O. Box 725 Bladensburg, MD 20710	Dave Hill P.O. Box 310-A Holland, MI 49423

==========

I would still like to publish 4 issues a year provided there is sufficient information. The delay in getting this issue out was due in part to a lack of enough material to make a comprehensive newsletter. Rather than send out a few measly pages, I wanted to wait until I could send something you would find informative and useful. With that in mind, instead of the subscription being for 12 months, it will be for 4 issues.

With the increase in the number readers, the cost of printing and mailing has gone up considerably. Therefore this will be the last of the complimentary issues. You will find subscriber information below.

As an added service to subscribers, I am making available all the programs and routines in the newsletter on 5-1/4 DSDD disc, or on tape if you have different drives. In the case of Tasword II, you must already own a copy of Tasword because what I will send you will be just the new machine code for XFER bytes and the new Basic. This will then need to be combined with the 10751 bytes of the Tasword II machine code. I don't want to step on any copyrighters toes. The cost of this service is $4.50. State whether cassette or 5-1/4 DSDD disc when ordering.

Mail to: The FD-68 User
P.O. Box 310-A
Holland, MI 49423

Cost: $15.00 for 4 issues
Programs: $ 4.50 per issue
Phone: (616) 335-8726

Make check or money order payable to: The FD-68 User

- -

Name_____

Address_____

City_____State_____Zip_____

Programs? Disc____ Cassette____ Total enclosed $_____

Type and Model of Drive(s)_____

_____

THE FD-68 USER
P.O. BOX 310-A
HOLLAND, MI 49423

GRAND RAPIDS, MI 495
AUG 1
PM
1986

Rod Gowen
1419-1/2 7th St.
Oregon City, OR 97045